Tensor Products for Cryptography: A Brief Paper

Philipp Harland

April 2025

Email: piranhafisherman@proton.me

Abstract

In this paper, we will explore some of the theory and uses of the tensor product in the cryptography and steganography. We will look over a few brief examples of methods using tensor products in practice in the duration of this paper.

Contents

1 Clarification **1**

2 Definitions; Theoretical Foundations **1**

3 Example in image-string steganography **2**

4 Examples in string cryptography **2**

5 Example in image-image steganography **3**

6 Example in audio steganography **3**

1 Clarification

By 'tensor product', we mean Kronecker product, or, $\otimes_{Kron}$. The definition itself, it will not be gone over in this paper. It is readily accessible on sites like MathWorld and Wikipedia.

2 Definitions; Theoretical Foundations

Def. 1. *The **horizontal tensor product** of 2 strings, S, and T, is the string, which is the result of replacing every i^{th} projection of $A_1Z_{26}(S)$, that is, $A_1Z_{26}(S)_i$, with $A_1Z_{26}(S)_iA_1Z_{26}(T)$. Then, we convert those values into letters by taking them mod 26, adding 1, and then mapping them to letters as their alphabetic*

values by Let: $\mathbb{Z}_{26} \to EngAlphabet$.
Def. 2. *The* ***matrix tensor product*** of two strings, X and Y, is the matrix A, where $A_{ij} = Let(A_1 Z_{26}(X)_i A_2 Z_{26}(Y))$. *This is a special case of an array tensor product, which takes an m dimensional array and an n dimensional array, to return an m+n dimensional array.*

3 Example in image-string steganography

We have an image which is an m×n array of $V_3(\mathbb{Z}_{256})$ *vectors, and, a string, S. We define the tensor product of these 2, by, first, taking the matrix tensor product of S with itself, and turning it back into an array of numbers via* $A_1 Z_{26}$. *Then, we take the 'tensor product' of the string as a matrix tensor product with itself and an image, w.r.t. each color channel. That is, for every value of every channel of each pixel, we replace it with itself scaling* $S \otimes_{Kron} S$. *S will be our 'key', and it is recommended for S to have a high 'complexity', and for it to be 'big', as in it is longer than 16 characters or so. This will make cryptanalysis much more difficult and resource consumptive.*

It will be more difficult and resource consumptive, because there will be more data points and with more diversity than if we used a smaller and less entropic key. Let's illustrate this – Let's say we have 2 identical row vectors, both of which are $(1,2,1)$. *Their (matrix) tensor product,* $(1,2,1) \otimes_{Kron} (1,2,1)$ *is* $\begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$. *Our attacker will only need a few guesses to achieve our original image, as there are 3 distinct entries – 1, 2, and 4. Our attacker will also need to use a minimal number of divisions (relative to the image's color channels), given there are 3 numbers, one being simply 1, which is the identity for multiplication. However, if we have 2 (identical) 'complex' strings, say, both being* $(3,4,2,8,2)$, *they would create a larger matrix with more diverse entries, which would clearly take more computational resources. This is even more so for 2 starkly distinct strings that are diverse in their entries.*

Here, we measure 'complexity' by how many unique letters are in the string. This is an elementary version of a 'string entropy' function.

4 Examples in string cryptography

We can construct a similar example to the one we created for image-string steganography, albeit using 2 strings instead of a string and an image – the string R will be the "key". the string J will be the "plaintext". R^2 *will be shorthand in this example for* $R \otimes_{Kron} R$. *We take the tensor product of J with* R^2, *or,* $J \otimes_{Kron} R^2$. *Here, we employ the same construction we did with the image example. Now, we are not technically encrypting a 'string' – but an array. Let's say Alice has a table of confidential data that she doesn't want Eve to access. We start by encrypting its elements, turning numbers into letters –*

albeit splitting them into individual digits and adding 1 to them all before doing so – and keeping the letters unchanged. Now, we have a string, S, which will be our 'key'. We take the tensor product of S with itself, S^2. Now, we take the tensor product of S^2 with every entry in the table, and, (optionally), remove any spaces by concatenating separated chunks.

5 Example in image-image steganography

For this, we have 2 images – I and K. I is our 'plainimage', and K is our 'key image', which is in grayscale. We first split I up into its color channels, and then apply the transformation, Y, to every vector – $Y(\begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix})$ *- which sets all* a_i *that are 0 to the one* a_j *that is nonzero, s.t. the scalar product of the vector with* $J_{1\times 3}$ *is* $3a_j$*. Now, we take the tensor product of each split up channel converted to grayscale with K, representing them as scalar fields. Now, we remove all projections except for the projection which corresponds to the channel they initially represented in the plainimage.*

6 Example in audio steganography

We have a signal, S, which has a "Fourier set" of pairs of frequencies and magnitudes, extracted from the frequency domain. We notate its elements $(\omega_i, \mathfrak{m}_i)$*. We have another signal, our 'key', with a same sized set of "Fourier pairs", which we notate* $(\Omega_i, \mathfrak{m}_{i,key})$*. We may take the tensor product by replacing* $(\omega_i, \mathfrak{m}_i)$ *with a set of pairs where* ω_i *is multiplied by every* Ω *in the key set, with the assigned magnitude being* $\mathfrak{m}_i\mathfrak{m}_{i,key}$*. If any 2 pairs coincide in frequency from this, we perturb one of them s.t. they are not equal in frequency to the other pair.* [1] [2] [3]

[1] Revision 1 – this is to correct 2 errors that slipped through proofreading, i.e. an index mistake in section 6 and a notation mistake in section 4.

[2] Revision 2 – Removed the 'Notes' section, which contained some claims that could be considered non-tentative. I also removed any non-tentative claims that did not have justification. Additionally, I added an argument for a claim in section 3.

[3] Revision 3 – Corrected a minor error in section 3.